hypre

High Performance Preconditioners

The solution of a large system of linear equations is the key computational bottleneck in many scientific and engineering applications. For example, the systems that arise in numerical simulation of three-dimensional radiation hydrodynamics may have upwards of 100 million unknowns. For problems of this size, one needs a scalable linear solver. In the context of preconditioned iterative methods, this means that the number of iterations required for convergence is independent of problem size. If this algorithmic scalability is coupled with a scalable implementation on a massively parallel computer, then the time to solution will remain constant as problem size increases with the number of processors.

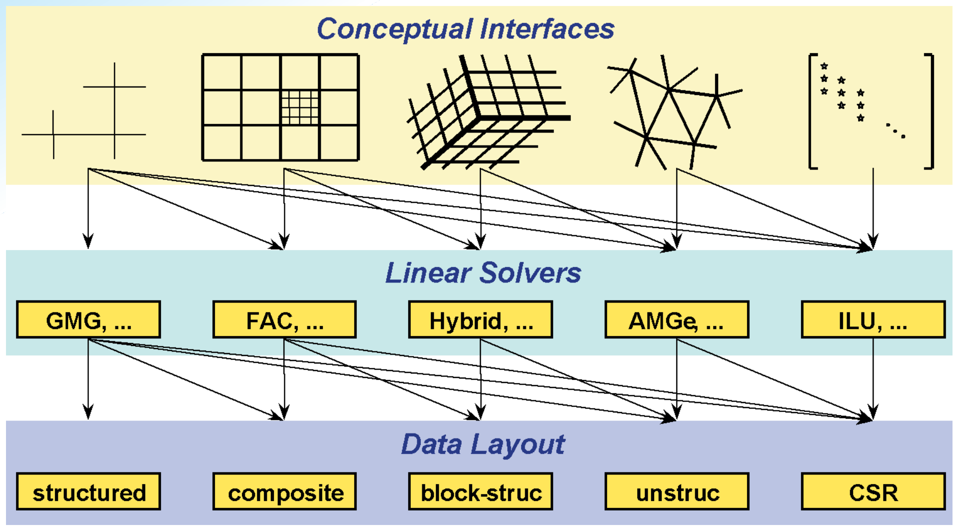

Figure 1: Conceptual interfaces are more natural for users, simplify coding, and provide access to better solvers.

CASC researchers in the Scalable Linear Solvers project are researching and developing methods for the scalable solution of large linear systems, including new parallel geometric and algebraic multigrid methods, incomplete factorization methods, and sparse approximate algorithms. The *hypre* project facilitates the use of these new algorithms in application codes through their development of the *hypre* library. The algorithms are implemented in a portable manner using standards such as MPI and OpenMP. Application interfaces are developed that ease the integration of *hypre* into complex application codes. Members of the *hypre* team work closely with key application customers, particularly those in the ASCI program (Advanced Simulation and Computing), to help them use *hypre* optimally within their codes and to gather requirements for further *hypre* development.

Conceptual Interfaces

Access to the linear solvers in *hypre* is achieved via conceptual interfaces (there are currently four). Figure 1 illustrates the idea behind conceptual interfaces. These interfaces (depicted in the top row of the figure) have a number of advantages:

- provide a more natural "view" of the linear system for applications.
- ease the coding burden for users by eliminating the need to map to rows/columns of a matrix (this can be extremely difficult in parallel).
- provide for more scalable "greybox" linear solvers.
- provide for more effective data storage schemes and more efficient computational kernels.
- provide polymorphic access to many of *hypre*'s solvers, i.e., application codes can experiment with different *hypre* solvers by changing only a single line of their source code.

The newest conceptual interface is the semi-structured interface, now implemented fully in parallel in *hypre*. This interface allows users to describe problems that are mostly structured, but with some unstructured features in them. For example, problems on block-structured grids (see Figure 2), structured adaptive mesh refinement (SAMR) grids, and overset grids can all be described via this new interface. The advantage is that the interface allows the underlying solver technology to take advantage of the structure that is present.

For instance, geometric multigrid ideas can be used, the primary computational kernels can be made very efficient, and storage can be minimized. In particular, several of the application codes (SAMRAI, ALPS, and AMRH) are using our current structured-grid solvers to do "level solves" in AMR applications. These level solves are usually embedded in an outer linear iterative solver such as the Fast Adaptive Composite (FAC) grid method. With this new interface, these solvers (i.e., FAC) can now be supported directly in the linear solver library.

One of the strengths of *hypre*'s design is that more general solvers are accessible through more specific conceptual interfaces. Thus, existing general matrix solvers BoomerAMG, PILUT, and ParaSAILS (Figure 3) are available through the more specific interfaces like the semi-structured interface. This concept extends to interoperability with other linear solver libraries such as those found in libraries like PETSc and Aztec: solvers in these

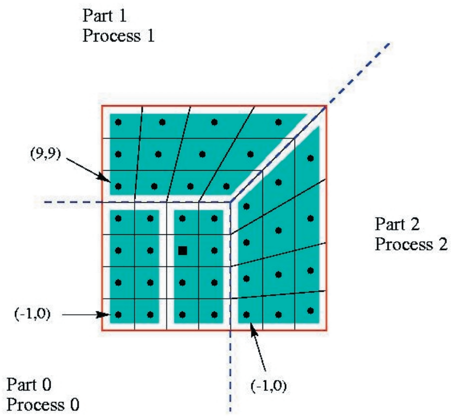

Figure 2: Semi-structured interface is used to describe problems that are mostly structured.

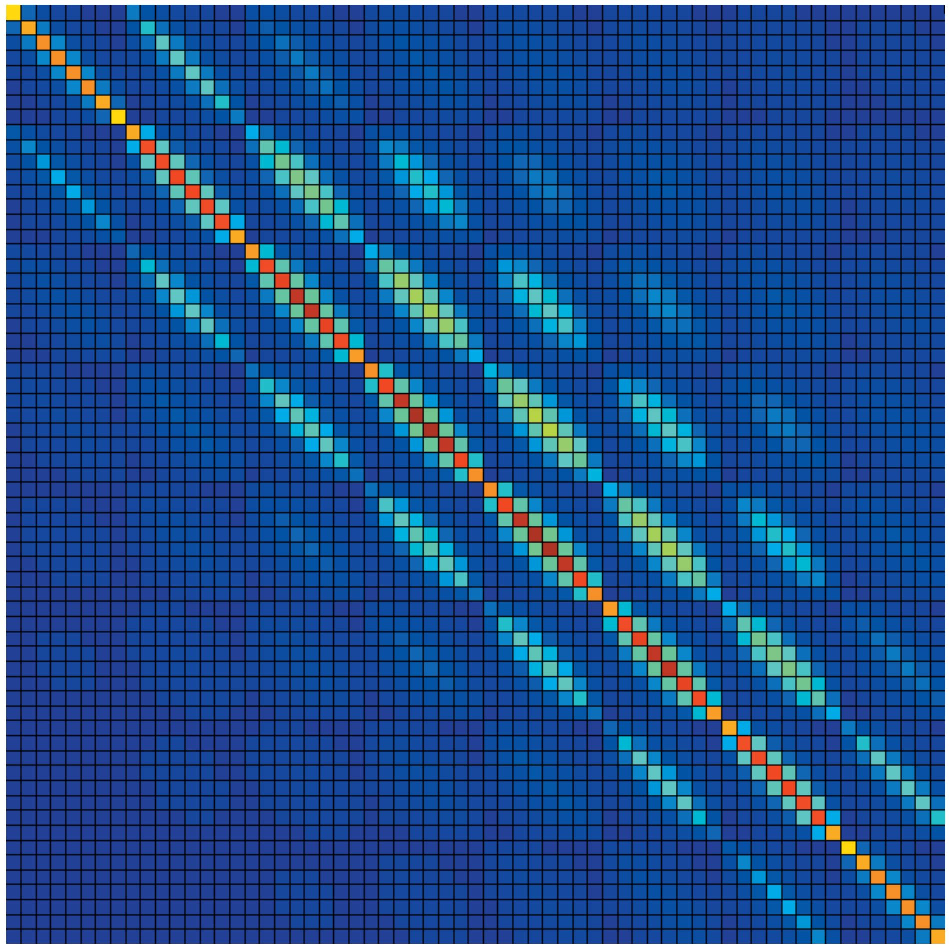

Figure 3: The exact sparsity pattern of the inverse of a sparse matrix shows that it can be well approximated by a sparse matrix.

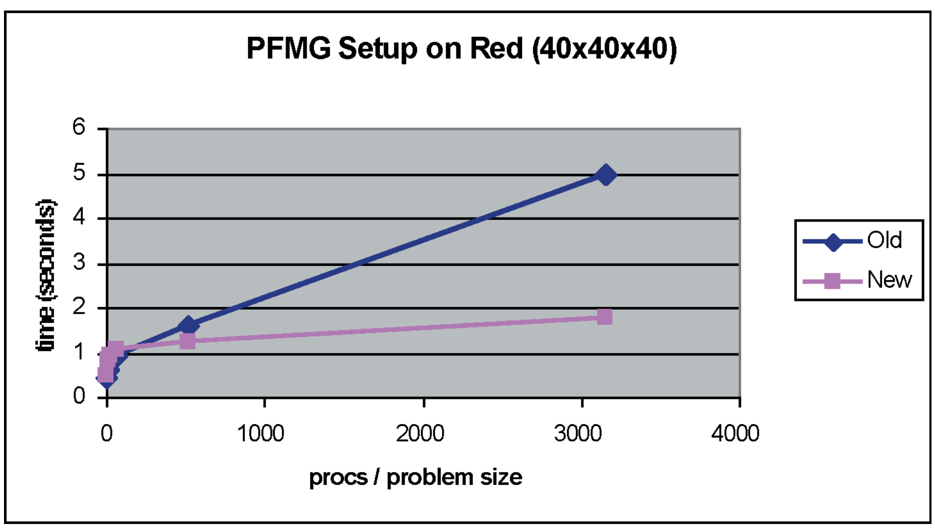

Figure 4: One setup phase algorithm scales like P, while a second algoithm scales like log(P).

libraries could be accessed through *hypre* if they shared the same conceptual interfaces.

Collaborations

Along these lines, the *hypre* project is collaborating with the CASC Components project, the ESI (Equation Solver Interface) forum, and the CCA (Common Component Architecture) forum, to develop standardized interfaces such as the semi-structured interface described above.

To further increase interoperability, the language interoperability tool "Babel" developed in the Components project is being used to provide basic object-oriented support and language interoperability to *hypre*. This will make *hypre* callable from many different languages, including C, C++, Java, Python, Fortran, and whatever new languages Babel supports. This is expected to be completed in 2002.

For more information contact Andy Cleary (925) 424-5890, or visit the Website at www.llnl.gov/casc/hypre.

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.